# Virtual Town Hall 2/19/2014

**Justin:** On behalf of the NIEM Program Management Office, welcome to the NIEM Virtual Town Hall! My name is Justin Stekervetz, Managing Director of NIEM. I'm especially excited to be with you all today as this is my first NIEM event since taking over the NIEM Managing Director position. So in kicking this off, we have a great technically-focused presentation on NIEM-UML. We'll also understand the business value behind the NIEM-UML specification and tooling implemented through it. And we'll also discuss the Design Patterns work that is scheduled to kick off very shortly. So let's get started and review the agenda.

Joining us for today's Town Hall is the state & local representative from the NIEM Technical Architecture Committee and Co-Chair, Mr. Andrew Owen, and our Lead Developer from Georgia Tech, Mr. Webb Roberts. After I provide you with the latest information on the NIEM-UML specification and tooling, Andrew will perform a demonstration on how to use the community-owned and open-source NIEM Modeling Tool. That particular tool leverages the UML profile for NIEM, commonly referred to as NIEM-UML, to generate XML schema for your information exchange need. Webb Roberts will then introduce you to our newest initiative around Design Patterns.

At this time, I would like to make note of a polling question on the left-hand side of your WebEx window. If you can take a moment to answer that particular question, it would be much appreciated.

So within the PMO, our role is to provide as much flexibility as possible when creating information exchanges. And that's why I am very excited to share an additional tool that has been added to the community toolkit. NIEM-UML is another great option that the community can use to more easily create exchanges in a visually oriented format. It's not a silver bullet, but it does have benefits for those who are used to working with UML.

So what is NIEM-UML? UML in itself, without the NIEM attached to it, is a model used to express designs. The standard arose from the Object Management Group. So when you add in the NIEM piece, you have a more powerful representation for designing and developing your NIEM-conformant information exchanges. So for those of you who understand UML, you might know that

objects exist, associations, packages, but the more fine-tune concepts such as subset schema, an extension schema, aren't represented in UML itself. What UML does is bypass those components and allow model-driven developers to develop conformant exchanges using tooling that supports the model.

NIEM-UML is about giving the community more options and expanding the NIEM community. It is not meant to be a silver bullet. That being said, if the Developer has experience with UML, NIEM-UML is certainly the tool that they would want to leverage. But by no means are we telling our NIEM XML schemas developers that they can't use their favorite XML schema development tool. That is still allowed, we're just providing another avenue for entry and providing a platform where business users might better understand the exchanges and the exchange requirements in a visual format.

NIEM-UML in and of itself is a specification; it's just a document. What makes it powerful is the tools that implement that particular profile. Today we have open-source and commercial tools, and those are found in our Tools Catalog.

So this means that architects and developers who build NIEM exchanges don't need to worry about the more specific details that are found in the NIEM Naming and Design Rules (NDR) because those details are presented at a much higher level. So you can create associations drawing the line between two objects, without having to understand the nuances of how you create associations in NIEM-conformant XML schemas. If you already use and understand UML, the typical learning curve that comes with NIEM development is greatly reduced. It is up to the community and/or private industry to provide that background knowledge of UML if the developer does not have previous experience. And there are plenty of trainings available out there for UML.

But I do want to stress at the end of the day, tooling that implements the profile will create NIEM-conformant schemas, so there is still a requirement for schema developers or implementers to be schema-aware in order to take the NIEM-conformant schema and implement it in that particular environment.

We know that some of you are already using NIEM-UML to create your exchanges or reverse engineer existing XML schemas that are NIEM-conformant. We would like your feedback. There are two easy ways to submit your experiences on the NIEM.gov website. First, if you would like to share your NIEM-UML story with the NIEM community, then simply click the "Submit New Project" button on the NIEM.gov/map webpage. And the links are noted on the slide. Otherwise, you can submit your NIEM-UML comments and lessons learned to the NIEM.gov "Contact Us" page. We'd certainly like to understand your stories, how you're leveraging tooling that implements NIEM-UML, and what capabilities you might want to possibly see in future versions.

So as many of you are aware, we recently launched the NIEM Version 3.0, and there were a number of enhancements that came with that version. Due to the fact that we voted on putting NIEM 3.0

in a 13-month consolidated timeframe, NIEM-UML lagged a bit behind in getting out to the street. So it's important to keep in mind when you're walking through the demonstration today, the tooling that you'll see implements NIEM Version 2.1, but hopefully in the next few months you'll see a Version 3.0 profile that developers will be able to take and implement with their particular UML tool.

And now that you have a good idea of what NIEM-UML is and what the value is, I'll hand the mic back over to the NTAC Co-Chair, Andrew Owen, who will demonstrate the Open Source Modeling Tool and can probably do a much better job than I can do. So I'll turn it over to you Mr. Owen!

**Andrew Owen:** Thank you, Justin, and very good foundation—everything you said. As I get my screen set up, I just want to reiterate a couple things Justin said. First and foremost, I'm really excited about the NIEM-UML Profile Version 3.0 because it gives us another entry point for folks interested in using NIEM. Usually you need to have an understanding of XML schema in order to develop exchanges and that's not the case anymore. Like Justin said, UML and XML are not mutually exclusive. If you're comfortable using XML schema, there's nothing stopping you from using that. The UML profile helps us attract a broader audience. What I'm going to demonstrate today is an open-source implementation of the profile. So as Justin said, this works with Version 2.1 of NIEM, with a version of 3.0 shortly to come.

So before getting into the Tool, I just wanted to introduce everybody to GitHub and tell everyone a little bit about what it is and how it relates to NIEM. GitHub is a commonly used location for managing source code and making software available to the broader community, primarily open-source. The tool I'd like to present today, like I said, is an open-source implementation of the NIEM-UML profile. What that means is that anyone interested in using the tool can do so as the tool is freely available, but also the source code is available if you are a developer or you have a developer that works for you that would like to add to the tool, make the tool better, stronger, and more feature-rich. The licensing allows us that as it is a completely open-source license. If you are not already a member of GitHub, I would recommend going out to GitHub.com and signing up. An account is all free and the link on the website will take you there. Once you do that, you can search for the NIEM-UML tool or you can simply go to the URL you see on my screen and then you can add to this project or submit issues or questions directly through GitHub and those questions or issues will go directly to myself or other individuals who are following or committing to this project.

If you are interested in downloading the source code and installing it in your environment, there are instructions here on how to build it, step-wise instructions. I hold this caveat that currently this is a little bit of a technical process, but we do have plans here short-term to assist and make it a little bit easier to download and walk through in 3 or 4 steps to install the software on your side. So that's a little bit on GitHub and where the tool is currently being hosted, and I know the NIEM PMO has plans for making GitHub a solution for other purposes.

So with that, I am now going to provide a brief demonstration of the open-source implementation of the NIEM-UML profile. Before I do so, I want to give a little bit more background on the tool itself and the method for developing the tool. UML is a highly adopted conformance standard. There is a lot of tool support and open source framework support for UML, so in the interest of time and saving money, we made a suggestion to build the NIEM-UML implementation in the Eclipse environment. If you are not familiar with Eclipse, what you see on your screen is what Eclipse looks like. Eclipse is a common development environment that software developers utilize and it really gives two big benefits. One is a common look and feel for the users. The layout will always stay the same whether I'm developing in Java or XML schemas or NIEM-UML; the general layout and the look and feel will be pretty much the same. The other big benefit is a large open-source community behind Eclipse. So what that means is when you set up to develop tools, you can make use open-source tools and frameworks that exist already. We didn't have to develop a UML tool, all we had to do was leverage a plug-in from Papyrus, which is a UML implementation in the Eclipse open-source community. We leveraged that tool and actually extended it to accommodate the NIEM-UML profile. As you may be aware, part of creating the NIEM-UML profile is to add extensions or customizations to UML to allow it to accommodate the concepts in NIEM. So if you are familiar with Eclipse, the rest should be fairly straightforward. If you don't, I don't think that's a big issue or big deal because it is very straightforward and something else to add to your list with GitHub. But the idea is once you download the Eclipse plug-in, you can just install it into your environment and from there you can start developing your NIEM-UML diagrams and build NIEM-UML specifications. Now what you see on the screen in front of you is a real example of a real NIEM-UML model. I say it's a real example, but it's just something I threw together for the scenario. Now in the interest of time and just understanding how live demonstrations can sometimes go, I prepared this information ahead of time so it's much easier to follow along and I'll make some modifications to it as I go along. But my primary goal is to give everyone a sense of the look and feel and what you're able to accomplish by using this implementation of the UML profile.

So I guess the first place we'll start is to understand that there are concepts in NIEM even though you'd think of them as XML-specific, if you abstract them enough they're not necessarily XML-specific. What I'm getting is when you look at a normal NIEM IEPD or a NIEM exchange specification you have things like an exchange schema, an extension schema, or perhaps multiple exchange schemas, as well as the subset schemas, a subset of the NIEM reference model. All of those concepts have found their way into the NIEM-UML profile as they are all critical to the functioning and operations of NIEM. So after I click on this left-most tab, what this gives me is a high-level view of what are the components of the existing model that I am developing. As you can see, these components consist of multiple things like a NIEM Core subset, an extension, an exchange. We have a Family Services subset. Family Services just so happens to be one of the many domains inside of NIEM. From here, what I have the ability to do is just double-click on either one of these and view the contents of that portion of the model, as well as remove or add some to that portion of the model. We have multiple diagrams here, but just understand these all are mutually supportive and they define a single model, and my intent is to produce a single IEPD, but because of the architecture of NIEM and therefore the architecture of the profile, we have multiple diagrams, one representing each namespace in the XML world.

So I guess I'm just going to start kind of from the top and work through this a little bit and talk through the specific components. As you may be aware, when you develop an IEPD, it's important to have at least a single root element. You can specify multiple root elements with your IEPD and define multiple messages. I would say the most common scenario is that you need to have a single root element and the NIEM rules say that when you define your root element at the exchange model level. So that's what we're doing here in this very simple diagram. The scenario here is that we need to exchange information about people. Very, very broad and very generic. But we have list of people, a collection of people, and we need to specify individual characteristics or properties of each person. So right here, we've specified this thing called "Person List" and Person List is intended to be the outermost root element of my information exchange. Understanding the way that UML works, this is saying here that Person List of Type:Person List is a substantiation of a class called Person List. So if you look up here, we have a class called Person List. So this would be equivalent to an XML type or XML complex type. So this defines the root structure for this exchange. In this case, because it's for example purposes, it's fairly straightforward. This exchange will contain information for one or more people, and the reason I know it's about one or more people is because I can refer to the cardinality in the annotation inside of my class. So we have the ability to easily manipulate that. I can select the attribute here and tweak that fairly easily. And you can see the property reflects that in the diagram. In this case I think it makes the most sense to leave it at one too many.

So the next level here is that we have the ability to full extensions to the NIEM model and actually pull in information from the NIEM reference schemas. Again, if you're more familiar with the XML side of this or at least the NIEM tooling side of this, think of the functionality currently in the NIEM Subset Generation Tool, that provides the ability to pick and pull pieces of the NIEM reference model that you need for your exchange purposes and add them to your IEPD. This tool provides very similar capabilities that ultimately the result in developing a subset. So I will jump into this tab called "NIEM core subset". And you can see I already have some information in here, again that is because I prepared a little bit before this session. As you can tell, we have some information in here about the vehicle, person, and then other information required up the hierarchy chain. There is also a couple of ways we can add to this diagram. First, let's say that I added this person type previously, but you know I need to add some information; I need additional information about a person so we can right-click on the class itself and choose "update subset." So, what is this going to do for us? This presents to us a tree view of the class we clicked on. In this case we clicked on "person type." This represents all of the available properties or characteristics of a person. And you can see that person access text—the first element here—is checked because I have already added it to my subset. Let's say I want to add a person's birth date. I need to choose how I want to represent that person's birth date, and I want to do so as a date, and I click update subset, and we see that the information is automatically added here. So again, intended to be very user friendly, somewhat I think represents the functionality the Subset Schema Generation Tool that folks may be familiar with today, just on top of UML as opposed to directly XML.

The other way that we can access the information in the NIEM reference schemas is through searching. So I just did a CTL F, but you can go up here to the search menu option. You see we have the ability to do searches. You we can search based on the full name of the element type, definitions, external concepts—NIEM 2.1 contains the concepts of external elements and types—you can search across those. So let's just say I need to add some information about an organization. I want to search across organization type. I am going to say full name only. The nice thing about this, the Eclipse tool set that we have used for a foundation of this has a lot of native built-in searching functionality, so we had to do very little in the way of enabling searching. So you will see at the bottom, search view, where all of the search results are displayed. And this is not specific to the NIEM-UML tool, but any piece of software you use in Eclipse this is where it displays the search result. So we were able to piggy-back on those capabilities and enable searching across the NIEM reference schemas. So what this shows me is that I have something called "organization type" in four different places in NIEM. So the search results are organized by their containing domain or their containing namespace and then you can actually see what that element is called under that namespace. So let's say for example, I just want to add the NIEM core organization type. I double-click on that and it's going to pop up a similar window as before. From here I can just pick what about an organization I want to add to my IEPD, to my model. Let's just say all we really need is the name of the organization, so we'll come out here and click "organization name" and when I do that you'll see that it automatically selects the parent type. I am going to choose update and now down here you'll see that in my NIEM core diagram it's added organization type. So that's really the extent of working with the NIEM subset. Our goal is to make this very simple, very clean, and very straightforward. To do so, we just wanted to make very few options in terms of how to add things to this subset. I guess I should show we do have the ability to remove things from this level as well. It is very common that once you develop the subset, you will want to add stuff to it, but also consider the scenario that you want to remove something from the subset. So it was a requirement at one point for the exchange, but now that thing is no longer needed. I'll use the person type as an example. If I just click on update subset like I did previously and say we don't need a person's accent anymore so I just click on that thing twice and you'll see the red "x" and I click update subset and that is no longer included—so it's no longer being specified as being part of my NIEM subset.

The other thing that is worth showing here is if you prefer just to drag and drop classes rather than doing the search feature, than you can certainly do that as well. Let's just say I want to add a class here. What we have done, because there are rules about inside of a NIEM namespace what elements can exist inside of those namespaces. That's really the value of NIEM is that you can grab a schema and know with certainty what is defined inside the NIEM core namespace or what's defined inside of the justice namespace, the immigration namespace. We have inserted checks in here so that users can't add invalid information to their subsets. So let's say, for example, I want to add some activity information. I click that, now I have my activity type. From here, I can simply right-click update and add information that would compose an activity. And we'll simply put the activity reason in there. And I know I said it before, but I think that pretty much covers the functionality in terms of building and managing a subset.

Next, I want to jump over to the final piece here, in terms of modeling, which is an extension. 99% of IEPDs that I have seen contain an extension, and it may even be higher than that. But, the point being that it is very common to have extensions to NIEM and be commonly referred as to as an extension schema. So in this case here, the scenario is that I need to share information about a person's favorite color. That's not something that is in NIEM. If you look at all the person information in NIEM, there is no way to convey a person's favorite color. So what I've done here is developed a new concept—a new class call it person—and I've added something called "person favorite color text." When I add a new element, I have to actually specify the types of that element so we know what the structure of that element is going to be. So you see here the person favorite color type has its own text type, and if you look down here in the properties view, you will see that text type is selected here, too. If I decide I want to change this to something, I have the option here and this gives me all of the available types in my model. Right now, there is nothing else in my model that would make sense to reassign this to. It's a text element, so I want to keep it as a text type, but if I wanted to change it to an instance to any other type, I can do so, via this dial in box. I will cancel out of that. The other important concept here is that I am creating a new concept called person, but it's really a specialization of something that's in NIEM already, so it's a special kind of something that is standardized in the NIEM model. We can determine this via the general association arrows that are on this diagram. In this case here, we have two person types. I just wanted to demonstrate the multiple levels of hierarchy we could have here. This person type, if you look down here in the properties view, this is from the Family Services subset, and I haven't opened that yet, but functionality on this side is very similar to what I demonstrated in the NIEM core subset. We have this thing called person type here, and person type contains person augmentation which has a tribal affiliation. So this allows us to convey tribal affiliation information for a person. You will see here that person type is a special kind of person type and this is the NIEM core person type. And that association is just carried over to my extension as well for convenience so I can look at this and understand that my person type extends the Family Services person type and that extends the NIEM core person type. So again, this is the very basics, I guess, of extending. The other thing I just wanted to point out here is that given the duration of this demonstration, we are keeping it at a very high level, so this tool has the ability to represent any of the concepts that are in NIEM. If you draw your attention to the right hand side here, the first two sections of nodes and edges, nodes and edges. These are all things that are default concepts in UML. However, as I said earlier in the call, as part of profiling UML for NIEM, we realize that there are concepts in the NIEM architecture that don't have a direct representation in the UML model. So we had to create our own concepts that extend the UML concepts. So if you look over here, you will see things like adaptors, and association types, augmentations, all that type of stuff, and all of these things are enabled and you can get access to those over here on the right hand side pallet.

Alright, I have about five minutes left, which is pretty good timing to talk about the final steps here. So, in most, if not all circumstances, the reason someone would be using this tool would be to produce a NIEM-conformant specification or an IEPD. Now, a big part of an IEPD would be the XML schema. In this tool, if you look to the NIEM-UML menu bar options, we have the ability to generate a NIEM MPD. If you do that, it will do exactly what I said. It will generate the MPD with

produces the XML schema. It takes a couple seconds, so I have done this previously. So you'll see in my project over here, I have a folder and this folder is just the name of my project. Because this is a demo, I have named this demo, but normally this would reflect the name of your IEPD or your exchange. And inside of this, we have the necessary information to specify the look and the feel or the structure of our message. So we have our exchange schema, and our extension schema—and I will get back to these in just a second here—we have our family services schema and our NIEM core subset schema. Now we did just add a few things to the model and since I didn't re-generate the XML schema some of that stuff isn't going to show here like the organization type and activity type, but the things that were in the model prior to the altering, are all reflected in the schema. The idea is that whatever you have in the UML model will result resulting XML schema. That is what we have here. So you'll see that we have a root element called personlist and list type personlisttype which is defined here. And that contains one or more person element, which happen to be defined in our extension namespace. We had to create a fake hold personfavoritecolortext because it is nc:texttype. We also have our person element here. And we can look and confirm proper organization of the NIEM core subset and the domain subsets as well. Other things worth noting is that this also produces a MPD catalog, which is a requirement of the MPD specification that allows tools to understand and work with the IEPD or the MPD. It also produces sample XML messages, which is a great way of allowing people to understand really the intended look and feel of the message. So this would really be the message that were to fly across the wire, as it is specified by the IEPD. And the last piece that I just wanted to introduce—and we won't have time to actually demonstrate it right now—but there is another piece of functionality in here: reverse engineering. So if you look to the NIEM-UML menu, along with generate NIEM MPD, is the option to generate an MPD model. And what this does is allow you to reverse-engineer to existing IEPD into UML. So as long as your IEPD has a conformant catalog file, that tells this tool where to find the extension schemas and the exchange schemas and the subset schemas, it will be able to produce the UML diagrams for you based on your IEPD. Alright. Well, that brings us to the conclusion of this demonstration. Again, I hope this was enough to give everyone a general sense of the goals, the intent, and really the capabilities of this tool, and broadly really understand that this tool has the ability to produce NIEM IEPDs that conform to the NIEM specifications, the naming and design rules, and the MPD specifications. I highly encourage everyone to take a look. It is available on GitHub: follow the directions for installation. If you have any questions, any areas clarity I think you can probably filter them through the NIEM Help Desk and get in touch with me or some of my staff one way or another. So I think I am handing it back over to your group now, Justin.

**Justin:** Thank you, Andrew. That is a great demo. And I would just like to remind participants that this tool aligns to NIEM version 2.1. NIEM version 3.0 is new in the process and we are currently working on a UML profile for 3.0. And we were providing introductory demo so we couldn't get into a lot of detail in order to ensure everybody had an opportunity at least see the tool and become comfortable with the tool. So, if you are working with the tool, GItHub is a great forum for posting questions and potentially providing feedback on how the tool could be enhanced. With that, and again thank you Andrew for that great demonstration. So with that, I am going to turn this over to Webb Roberts. He is going to talk about NIEM Design Patterns: what they are and how you as the NIEM community can get involved. Webb, are you ready?

**Webb:** Yes, I am.

**Justin:** Great.

**Webb:** My name is Webb Roberts. I am an employee of the Georgia Institute of Research Technology (GTRI). We have been working on NIEM since the beginning. I myself am the author of the naming and design rules—well, the principal author—and NIEM high level version architecture. We are here today to talk about a new effort for NIEM that we are calling NIEM Design Patterns. A NIEM Design Pattern is a repeatable pattern or template for software development that can be applied in different environments or with different technologies. Our goal here is to enhance and encourage integration of NIEM with different technologies, software system, and languages, including query and programing languages. We want to ensure we are making the last mile. It's not just about developing exchanges, but it's about building data from real system data, about being able to integrate these exchanges into existing software and integrating NIEM data definitions and exchange NIEM data with existing technologies. And to take these methods and publish them as reusable Design Patterns so that other people can take advantage of them. What we are looking for is for developers who have existing capability related to NIEM for information exchange and have a concrete need to integrate NIEM with their existing software that have a real capability to use the technology with a need to integrate to integrate NIEM with that technology—and I am going to call this an "integration need". We are looking for volunteers who want to work with us to develop a new method or a process for integrating NIEM with their systems. We will work with these volunteers to define a solution for integrating NIEM with their technology. We're also looking for people who have defined innovative solutions for integrating NIEM with common software systems. So if you are ahead of the curve, we would like to hear from you so that others can be helped using the same methods you created. For example, if you are using software systems that use other existing exchange languages like RDF and OWL or JSON or other XML variants, integrating with databases including relational databases and or SQL databases, using query languages such as RDF and OWL or SQL, or programming languages including JAVA, JAVA script, SCALA, python, BASH, PHP. These are just examples of integration points where NIEM data and NIEM data definitions need to integrate to your existing software. So if you have an integration need to work with NIEM and to work with your existing capability, talk to us and let's see what we can do to help you and to help the entire community build on what you have done.

The steps in this process are first we are engaging the community. We are looking for integration needs and volunteers who will help us develop them into these repeatable methods that we're calling Design Patterns. Then we are going to work with volunteers to find solutions to defining solutions to the integration needs and formulate them as Design Patterns. Then we will publish them as a publicly viewable online catalog: the Design Patterns catalog. These steps—one through three—are going to be conducted in parallel, so we will not hold back the entire Design Patterns Catalog until all of the Design Patterns are finalized. We'll see them trickle out as they are defined and as they are finalized. Then we will build from one of these Design Patterns a candidate specification. The goal here is to define a method using one of these Design Patterns that will be

considered NIEM-conformant. For example, a spec that might come out is a NIEM-conformant JSON exchange. The specification will be vetted through the NIEM Technical Architecture Committee and defined as an official specification to be released on reference.niem.gov. And the goal is to define a to be considered as NIEM-conforman as NIEM-XML is right now. So, how do we get involved? Go to NIEM.gov/contactus—all one word—contactus. Tell us who you are and what your integration need is. And we will get in touch with you to see how we can help. If you would like to stay updated you can go to NIEM.gov, look under the menu item "Design Patterns." This will point you to our current status and how this is going. We look forward to hearing from you. Thank you.

**Justin:** Thank you, Webb, for that overview of the effort. And from the PMO, we are very excited to get this started and certainly welcome your participation. It's necessary, so we're looking at requirements and needs of the community and not coming up with them ourselves. You are the boots on the ground performing the exchange work and implementing them within your environments.

**Christina:** We did have some questions that came in throughout the course of the Town Hall. The first question will be asked towards you. In the exchange diagram, in the demo that you provided, what is "propertyholder"?

**Andrew:** That is a good question. Normally I make it a point to touch on that during the demonstration. One of the issues we has to address when understanding how to represent NIEM or not just NIEM I guess, but represent XML concepts in UML is the idea of a globally declared data element. So what I mean is as an element is declared at the utmost level of, the top level, of a schema is not encapsulated inside of another type. Actually, the NIEM naming and design rules require that all elements are defined globally. The problem in mapping this to UML is there is no clear way to globally declare elements in UML. So the profile introduces this concept of a property holder, which is a stereotype applied to a UML class. And basically, the rule says any element, or any attribute, I guess—that is declared inside of a property holder class would equate to a globally declared element in NIEM. So another way to look at is that when I press that button to produce a NIEM-conformant IEPD, the transformation that does that work lists the property holders and writes pulls those attribute out as globally declared elements in XML.

**Christina:** Thank you, Andrew. Webb, the next question is coming to you. What type of information do you need from me if I have a Design Pattern business need and also what will my responsibilities be if I volunteer such as time commitment?

**Webb:** What we need from you is an understanding of what your capability is and what your need is and then what you're willing to volunteer. So if you are capable of working with this particular technology, have an existing capacity, and you need to integrate NIEM, let us know what the existing capacity is and what the shortcoming is. So, what is the bridge we need to build? Even if you can't volunteer, then let us know what that need is. Even if you can't give us time, perhaps

we will be able to try to build out a solution. But, having you as a volunteer, willing to work with us to vet and test the Design Patterns. One, that is going to make it a higher priority because we are going to be able to actually realize a result from this and two, it will make it will improve the quality of the output. So, even if you can't become a volunteer, please let us know what the shortcomings are and how you can help. Thanks.

**Christina:** Thanks, Webb. Andrew, it's coming back to you. Does the NIEM Modeling Tool provide a quality check to, for example, make sure a NIEM class is not added to an extension package?

**Andrew:** There certainly are quality checks in there. For example, if you try to add something to a NIEM package that is not allowable based on the NIEM reference model, you wont be allowed to do that. The reverse of that is say you were to add something called "persontype" to your extension namespace, and "persontype" obviously exists in the NIEM core as well and the tool is not going to argue about that because that is an allowable activity. One of the rules that has been a long standing golden rule of NIEM is that you don't want to invent something that already exists in NIEM. That is not to say that you can't have a type in two namespaces with the same name, but if you have something that is called "persontype" in your extension schema that should be because is it a specialization or extension of the persontype that is in NIEM core or in the other NIEM namespaces. So, quality from that perspective is really up to the user to make sure they are not replicating things in NIEM. But we are able to control and check to make sure that the information entered in the NIEM reference models is accurate.

**Christina:** Thank you for that, Andrew, but don't go too far because you have the question as well. Does the tool use the Subset Generation Tool known as SSGT?

**Andrew:** It doesn't use it, no. The SSGT is a tool that has similar capabilities to what this tool does. But that's a web offering that is really based on users browsing the website and building the subset. These capabilities are built into this tool as well, but it does make direct use of the SSGT.

**Christina:** How does this tool deal with codelists and enumerations?

**Andrew:** Two things. There is the concept in UML of enumerated lists. So, if you were to search through the NIEM subset, for example, and find codes that exist in the NIEM model today—things like hair color, eye color, other demographics—those things are represented in here. The other side of it is if you need to create your own codelists, you can certainly do so. That was I guess one of the things that was in the lower right corner tool panel: the ability to add enumerations and then add literals to those enumerations. Now, the one thing that I'll say we don't have in here yet that I know is of interest to the community is the ability to do some sort of automatic import—if the codes are in an Excel spreadsheet, for example, is there a way to auto-import those into this tool? The answer to that is right now, not cleanly, but since UML is really serialized as XML under the covers, you have the ability to use transformation in your own environment. Since this is an open-source tool, the source code is freely available and if users just have an interest in that, we would love to see that tradition.

**Christina:** Ok thanks, Andrew. Justin, the next question is for you. What is the need for UML when we have NIEM-compliant IEPDs?

**Justin:** The NIEM-conformant IEPDs are really output of a tool that implements NIEM-UML. As I mentioned earlier, we're not asking or requiring people to replace their favorite XML schema development tool with UML. We're trying to widen the audience, bring modelers to the table who have a preference for UML or maybe those who are brand new to NIEM, don't know XML schema, don't know UML, but prefer a visual approach. So, it is not replacing IEPDs, what it's doing is providing capability to produce NIEM-conformant IEPDs leveraging a visual model, which is the Unified Modeling Language. Again, it does still require some level of knowledge of XML schema, because the output is XML schema and you the assumption is you will have to implement your IEPDs, so you'll have to have some knowledge about how to implementing it within your particular environment. Andrew, do you want to add anything to that?

**Andrew:** No, I think you covered it well.

**Christina:** Andrew, the next question is coming your way. Does the NIEM Modeling Tool generate wantlist.xml?

**Andrew:** No, it does not. And I can elaborate a little bit on that. The wantlist.xml is really specific to the XFPT that we talked about. I believe the facing initially if you are developing your in the tool, it is certainly not a bad idea and something we could consider adding down the road.

**Christina:** Great. Justin, the next question is for you. Has the NIEM-UML specification been tested in any commercial proprietary environment other than Eclipse?

**Justin:** So, we have the open-source tool, that this does require those who produce UML modeling tools to create plugins or embed NIEM-UML into that particular tool. As the NIEM-PMO, we are providing these specifications as we provide for folks to build tools against. We do not know of another commercial project from No Magic that has a plug in. There are likely others out there that we don't know about, so if you are aware of anything outside of Eclipse, No Magic, please share that information with the community. But again, we provide these specifications as a mechanism to build against and allow the community to allow the tooling to flourish in their particular communities of interest.

**Christina:** I believe we have one more question.  Justin, this is geared towards you. For the Design Patterns initiative, is this a one-time deal or is it going to be a continuous effort going forward?

**Justin:** I think the program prefers continuous engagement with the community. The program wants to stay on the edge, not the bleeding edge, but the edge where we're making requirements that exits and building towards them. Not moving faster than the community is looking for us to move. So, I would assume that we would still continue to take input and our Technical Architecture

Committee to consider the pattern and potentially build a specification against it, should it be warranted. Webb, do you have to add anything to that?

**Webb:** We're trying to get a good start on building a Design Patterns catalog and I think once that is stood up, we can use that as the basis for additional Design Patterns down the road.

**Justin:** Great. Thanks, Webb. So, that is the final question. I would really like to thank everybody for their participation today. Hopefully you learned about the UML profile for NIEM and in particular, the NIEM Modeling tool, which implements it as well as Design Patterns. And I hope you're excited and looking to submit your suggestions or indicate your participation in the effort because this is a community driven effort—just like NIEM 3.0 was—we do need your help in order to better understand your requirements and chart the path forward. So with that, thank you everybody, we appreciate your time, and until the next Virtual Town Hall.